Note that no dedicated wires or registers are required to transmit the address field. It is transmitted using the data bits. As is explained below, a pipeline stage will not be slowed down if it is not intended to be activated by the particular address field, i.e., the stage will be able to pass along the token without delay.

The remainder of the data in the token following the address field is not constrained by the use of tokens. These D-data bits may take on any values and the meaning attached to these bits is of no importance here. That is, the meaning of the data can vary, for example, depending upon where the data is positioned within the system at a particular point in time. The number of data bits D appended after the address field can be as long or as short as required, and the number of data words in different tokens may vary greatly. The address field and extension bit are used to convey control signals to the pipeline stages. Because the number of words in the data field (the string of D bits) can be arbitrary, as can be the information conveyed in the data field can also vary accordingly. The explanation below is, therefore, directed to the use of the address and extension bits.

In the present invention, tokens are a particularly useful data structure when a number of blocks of circuitry are connected together in a relatively simple configuration. The simplest configuration is a pipeline of processing steps. For example, in the one shown in Fig. 1. The use of tokens, however, is not restricted to use on a pipeline structure.

Assume once again that each box represents a complete pipeline stage. In the pipeline of Fig. 1, data flows from left to right in the diagram. Data enters the machine and passes into processing Stage A. This may or may not modify the data and it then passes the data to Stage B. The modification, if any, may be arbitrarily complicated and, in general, there will not be the same number of data items

Boolean algebra, it can be shown that the output signal SA from this logic block (the output from NOR1) is HIGH (a "1") only when the previous extension bit is a "0" (QPREV="0") and the data word at the output of the non-inverting Q latch (the original input word) LDIN has the structure "000001xx", that is, the five high-order bits MD[7]-MD[3] bits are all "0" and the bit MD[2] is a "1" and the bits in the Zero-one positions have any arbitrary value.

There are, thus, four possible data words (there are four permutations of "xx") that will cause SA and, therefore, the output of the address signal latch LADDR to whose input SA is connected, to become HIGH. In other words, this stage provides an activation signal (DATA_ADDR = "1") only when one of the four possible proper tokens is presented and only when the previous extension bit was a zero, that is, the previous data word was the last word in the previous series of token words, which means that the current token word is the first one in the current token.

When the signal QPREV from latch LEPREV is LOW, the value at the output of the latch LDIN is therefore the first word of a new token. The gates NAND1, NAND2 and NOR1 decode the DATA token (000001xx). This address decoding signal SA is, however, delayed in latch LADDR so that the signal DATA_ADDR has the same timing as the output data OUT_DATA and OUT_EXTN.

Fig. 7 is another simple example of a state-dependent pipeline stage in accordance with the present invention, which generates the signal LAST_OUT_EXTN to indicate the value of the previous output extension bit OUT_EXTN. One of the two enabling signals (at the CK inputs) to the present and last extension bit latches, LEOUT and LEPREV, respectively, is derived from the gate AND1 such that these latches only load a new value for them when the data is valid and is being accepted (the Q outputs are HIGH from the output validation and acceptance latches LVOUT and LAOUT,

Two additional interrupt service routines are required:

- accept enable interrupt
- Target met interrupt

When a target met interrupt occurs, the service routine should add an enable to its off-chip enable queue.

A.12.7.1 Output gate logic behavior

Writing a 1 to the enable_stream register loads an enable into a short queue.

When a FLUSH (marking the end of a stream) passes through the output gate the gate will close. If there is an enable available at the end of the queue, the gate will open and generate an accept_enable_event. If accept_enable_mask is set to one, an interrupt can be generated and an enable is removed from the end of the queue (the register enable_stream is reset).

However, if accept_enable_mask is set to zero, no interrupt is generated following the accept_enable_event and the enable is NOT removed from the end of the queue. This mechanism can be used to keep the output gate open as described in A.12.5.

A.12.8 Bit counting

The bit counter starts counting after a FLUSH Token passes through it. This FLUSH Token indicates the end of the current video stream. In this regard, the bit counter continues counting until it meets the bit count target set in the bit_count_target register. A target met event is then generated and the bit counter resets to zero and waits for the next FLUSH Token.

The bit counter will also stop incrementing when it reaches it maximum count (255).

A.12.9 Bit Count Prescale

In the present invention, $2^{(bit\_count\_prescale+1)}$ x 512 bits are

state MB4A:

```
vmb_addr = vmb_addr + blocks_per_mb_row;
new_state = DATA;
```

state (MB4) and MB4B:

```
(scratch = hmb_addr - last_mb_in_half_row;)
if (z & (mod3==2)) /*end of slice on left of screen*/
{
   hmb_addr = hmb_addr + maxhb;
   new_state = MB4C;
}
else if (z) /*end of row on left of screen*/
{
   hmb_addr = 0;
   new_state = MB4A;
}
else
{
   hmb_addr = hmb_addr + maxhb;
   new_state = DATA;
}
```

states MB4C and MB4D:

```
vmb_addr = vmb_addr + blocks_per_mb_row;
vmb_addr = vmb_addr - blocks_per_mb_row;
new_state = DATA;
```

states MB5and MB6:- as above

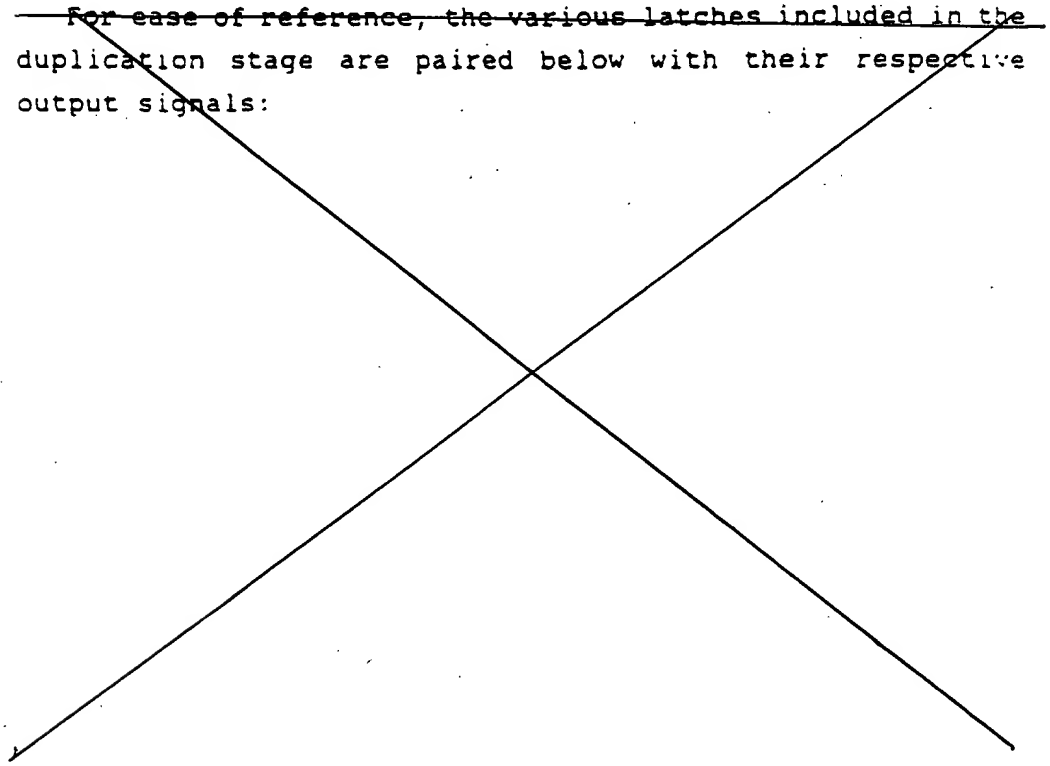
C.3.5.3 Operation on PICTURE_START Token

When a PICTURE_START token is received, control passes to state PIC_ST1 where the vb_addr register(BU_WADDR_VBADDR)is reset to 0. Each of states PIC_ST2 and PIC_ST3 are then visited, once for each component, resetting hmb_addr and vmb_addr respectively. Control then returns, via state OUTPUT_TAIL, to IDLE.

interface according to this embodiment can be adapted very easily to different applications.

The duplication stage shown in Fig. 8 also has two latches LEIN and LEOUT that, as in the example shown in Fig. 6, latch the state of the extension bit at the input and at the output of the stage, respectively. As Fig. 8a shows, the input extension latch LEIN is clocked synchronously with the input data latch LDIN and the validation signal IN_VALID.

For ease of reference, the various latches included in the duplication stage are paired below with their respective output signals:



In the duplication stage, the output from the data latch LDIN forms intermediate data referred to as MID_DATA. This intermediate data word is loaded into the data output latch LDOUT only when an intermediate acceptance signal (labeled "MID_ACCEPT" in Fig. 8a) is set HIGH.

The portion of the circuitry shown in Fig. 8 below the acceptance latches LAIN, LAOUT, shows the circuits that are added to the basic pipeline structure to generate the various